



CONSEJERÍA DE EDUCACIÓN
Comunidad de Madrid

IES ENRIQUE TIERNO GALVÁN
Parla

**CFGS DESARROLLO DE APLICACIONES
MULTIPLATAFORMA**
Curso 2024/2025

Proyecto DAM

TÍTULO: Aplicación gestión de videojuegos.

Alumno: David Bargalló Ortiz.

Tutor: Julián Parra Perales.

Junio de 2025

Índice

Contexto de la aplicación	3
Mundo real del problema	3
Qué aplicaciones existen.....	3
Mejoras de la aplicación respecto a las existentes	3
Casos de Uso	4
Requisitos funcionales y no funcionales	7
Funcionales.....	7
No funcionales.....	8
Diseño.....	8
GUI.....	8
UI - User interface	8
UX - UserXperience	11
Diagrama navegación	12
Arquitectura	13
Diagrama	13
Componentes	13
Base de datos	14
Plan de pruebas.....	15
Pruebas para el registro (Clase UsuarioServicio):	15
Pruebas para inicio de sesión (Clase UsuarioSevicio):	15
Pruebas sobre los videojuegos en la App (Clase VideojuegoServicio):.....	15
Pruebas sobre los videojuegos de un usuario específico (Clase Usuario):	16
Pruebas para guardar la compañía (clase CompaniaServicio):	16
Pruebas para guardar una ubicación (clase UbicacionServicio):.....	17
Pruebas de los filtros para los juegos en la bbdd (clase FiltrarVideojuego):	17
Pruebas sobre la paginación (clase Paginador):.....	17
Pruebas de conexión (no unitarias):	17
Implementación	17
Entorno de Desarrollo	17
Implantación/Puesta en producción.....	18
Capturas en ejecución de la aplicación (funcionalidades)	18
Información sobre la versión y software necesario	25

Elementos destacables del desarrollo.....	25
Manual de usuario	26
Inicio de sesión:.....	26
Registro de usuario:	27
Biblioteca de juegos:	27
Búsqueda de juegos para añadir:.....	30
Wishlist:.....	33
Estadísticas:	34
Conclusiones	34
Bibliografía	37
Anexos	38

Contexto de la aplicación

Mundo real del problema

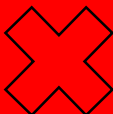

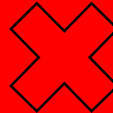
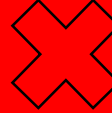
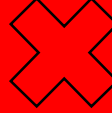
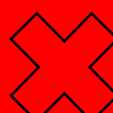
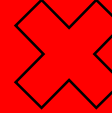
Muchos jugadores acumulan videojuegos en diferentes formatos y plataformas, lo que dificulta su organización. Esta aplicación busca centralizar toda esta información en un único lugar, resolviendo problemas como la desorganización, dificultad de localización, y falta de estadísticas.

Qué aplicaciones existen

Actualmente existen algunas aplicaciones que ayudan a gestionar colecciones de videojuegos, como GG App, HowLongToBeat o Backlogg. Estas aplicaciones permiten realizar un seguimiento, pero no ofrecen funcionalidades como el almacenamiento físico. Además, todas son aplicaciones web, esta es una alternativa a nivel de aplicación para la gente que prefiera tener la aplicación a mano, y que no sea como puede ocurrir con las otras, que un fallo en la web puede evitar que accedes a tu biblioteca durante mucho tiempo. Con la solución que ofrecemos con esta aplicación, solo se necesita conexión a internet y que la web donde se aloja la base de datos funcione, no dependes de las webs del servidor de la empresa o posibles ataques a la misma.

Mejoras de la aplicación respecto a las existentes

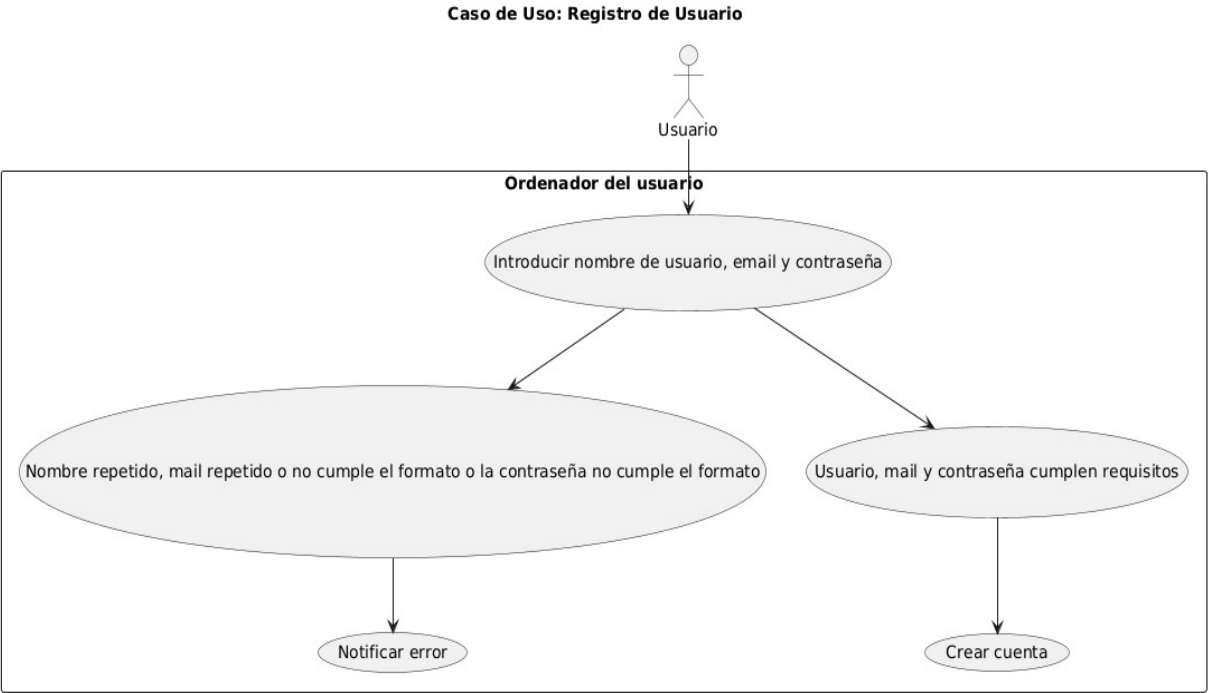
La aplicación permite localizar juegos físicos, mostrar estadísticas personalizadas (sobre nosotros, como los géneros más jugados, consola para la que más juegos tienes...), además de funcionalidades como exportar colecciones en PDF y conectar con bases de datos en la nube. Y como mencionamos anteriormente, no dependes tanto de factores externos como el servidor de una web.

Aplicaciones	Localización de juegos físicos	Estadísticas personalizadas	Exportar biblioteca	Dependencia de factores externos	¿Una solución desktop?
GG App		Parcialmente (estadísticas generales)		Servidor web, ataques externos...	Ofrece versión desktop, pero se centra en la web
HowLongToBeat		Solamente sobre duración de juegos.		Servidor web, ataques externos...	
Backlogged		Parcialmente (estadísticas generales)		Servidor web, ataques externos...	Ofrece una versión desktop, pero se centra en la web.
Biblioteca de los Videojuegos	Lo tiene.	Gráficos con diferentes estadísticas	Ofrece la funcionalidad de exportar	Sin depender de servidores de terceros,	Full desktop, con funciones más concretas y

		(géneros, consolas...)	la biblioteca en PDF.	más difícil recibir ataques (y poner en peligro tus datos)	menos generalistas.
--	--	---------------------------	--------------------------	--	------------------------

Casos de Uso

- Registro



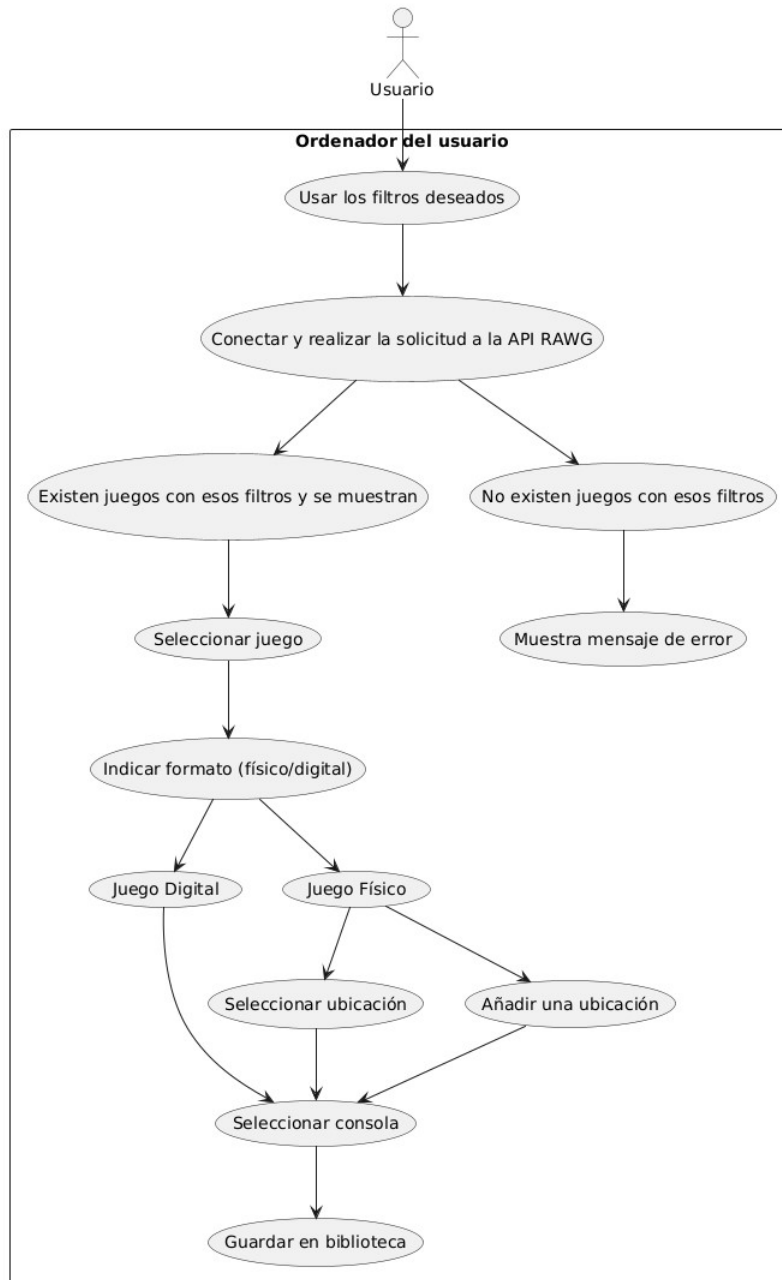
- Filtrar juegos en biblioteca

Caso de Uso: Filtrado de Juegos en Biblioteca



- Buscar juego para añadir

Caso de Uso: Buscar Juegos para añadir a la biblioteca



- Exportar biblioteca a PDF

Caso de Uso: Exportar Biblioteca a PDF



Requisitos funcionales y no funcionales

Funcionales

- El usuario podrá autenticarse introduciendo el nombre de usuario y la contraseña con la que se registró.
- El usuario podrá buscar videojuegos tanto los que están en su biblioteca, como nuevos juegos para añadirlos a la misma, usando diferentes filtros para ello.
- Permitir guardar, añadir y eliminar videojuegos.
- Tener una wishlist propia.
- El usuario podrá tener una pantalla con estadísticas basadas en su biblioteca.

- Podrá exportar su biblioteca y su wishlist en formato PDF.
- Cada usuario tiene su propia biblioteca y wishlist separadas.

No funcionales

- Conexión a servicios externos que proporcionan la información de los juegos.
- Conexión a la BBDD en línea.
- Interfaz sencilla e intuitiva.
- Contraseñas almacenadas y protegidas en la base de datos.
- Aplicación a la que se le pueden añadir nuevas funciones de forma sencilla.

Diseño

GUI

UI - User interface

Vamos a comentar con los bocetos para el diseño de las siguientes pantallas que tendrá nuestra aplicación: login, registro, biblioteca, búsqueda, wishlist, estadísticas, detalle del juego.

Pantalla login:

TITULO DE LA APP

NOMBRE USUARIO

CONTRASEÑA

INICIAR SESIÓN REGISTRARSE

Pantalla registro:

PANTALLA REGISTRO

NOMBRE USUARIO

CORREO ELECTRÓNICO

CONTRASEÑA

REPETIR CONTRASEÑA

REGISTRARSE

VOLVER A
INICIO SESIÓN

Pantalla biblioteca:

PANTALLA BIBLIOTECA

CONSOLAS:

NOMBRE:

GÉNEROS:

ORDENAR POR:

APLICAR FILTROS

FICHA DEL JUEGO

PORTADA

NOMBRE: JUEGO
CONSOLAS: CONSOLA1
GÉNEROS: GÉNERO1
EMPRESA: EMPRESA1

ELIMINAR

<----- PÁGINA X DE X ----->

AGREGAR JUEGO

Pantalla búsqueda juegos:

PANTALLA BÚSQUEDA

CONSOLAS:

NOMBRE:

GÉNEROS:

ORDENAR POR:

BUSCAR JUEGOS

FICHA DEL JUEGO

PORTADA

NOMBRE: JUEGO

AÑADIR A BIBLIOTECA

AÑADIR A WISHLIST

<-----

PÁGINA X DE X

----->

Pantalla estadísticas:

PANTALLA ESTADÍSTICAS

CONSOLA CON MÁS JUEGOS: CONSOLA 1

GÉNERO MÁS JUGADO: GÉNERO 1

TOTAL DE JUEGOS FÍSICOS: X

TOTAL DE JUEGOS DIGITALES: Y

GRÁFICOS DE LAS ESTADÍSTICAS:

PIECHART



GRÁFICO DE LÍNEAS



Pantalla wishlist:

PANTALLA WISHLIST



JUEGO 1

TIENDA 1

TIENDA 2

AÑADIR A BIBLIOTECA



JUEGO 1

TIENDA 1

AÑADIR A BIBLIOTECA

Ficha juego:

FICHA DE JUEGO

PORTADA

JUEGO: JUEGO1

CONSOLA: CONSOLA1

GÉNEROS: GÉNERO1, GÉNERO2

FECHA DE LANZAMIENTO: XX-XX-XXXX

FORMATO: FISICO/DIGITAL

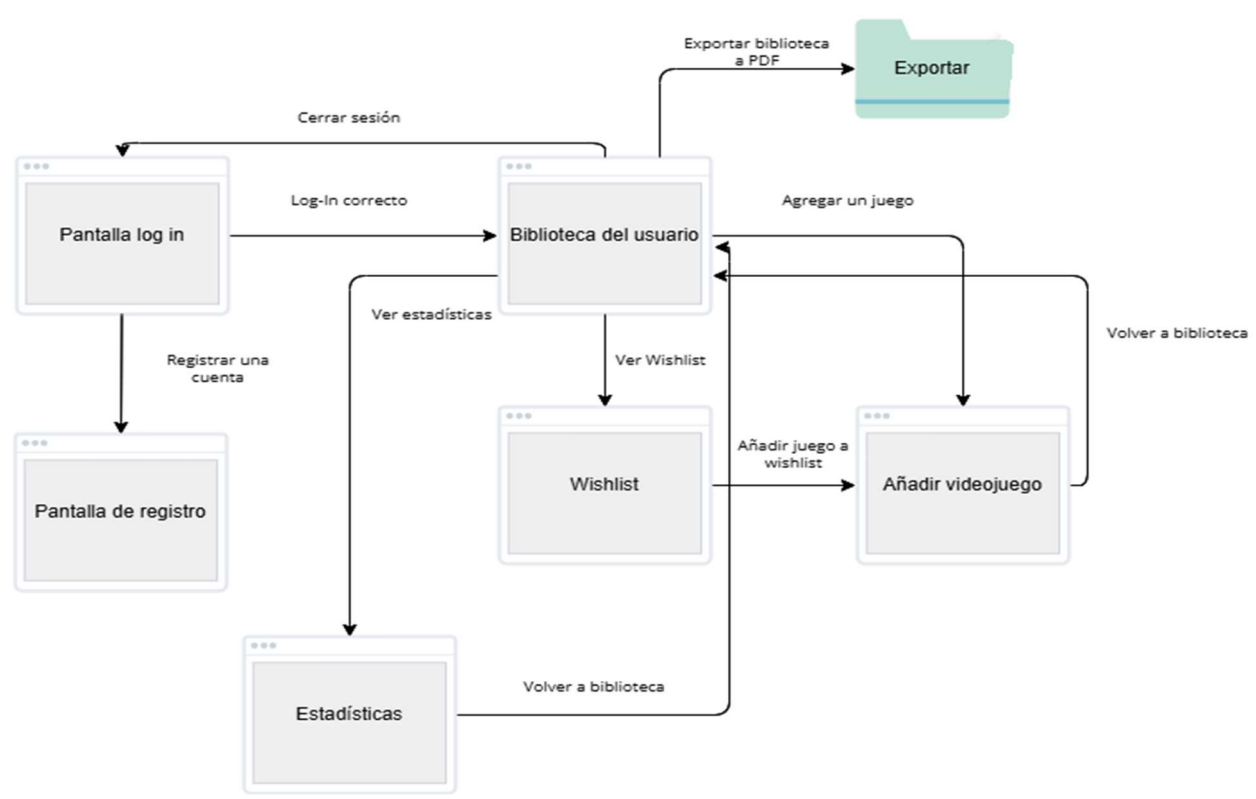
UBICACIÓN: UBICACIÓN1/NA

CERRAR

UX - UserXperience

Diseño sencillo, colores agradables a la vista para el usuario y combinación de fondos y colores legibles, la mayoría de funcionalidades son muy explícitas, y se pueden usar de manera sencilla sin necesidad de indagar mucho. Se ofrece la mayor comodidad al usuario, ofreciéndole siempre que se puede opciones para que él solamente tenga que pulsar la opción que necesite, o incluso en algunos casos, poder añadir él una opción manualmente desde la misma pantalla (como la ubicación física). También, en el tema de filtrado, para mayor comodidad, la mayoría de campos en los que es posible, se le ofrece un desplegable con todas las opciones, para así evitarle el escribir cada campo a mano, solamente deberá escribir el nombre del juego, e incluso puede escribir una parte del nombre, que saldrán los juegos que coincidan con lo que ha escrito, muy útil si no recuerda el nombre de un videojuego que desea añadir. En cuanto a la búsqueda de juegos, se le ofrece una gran biblioteca de juegos actualizada a la última para añadir dichos juegos en su biblioteca o wishlist según desee. La wishlist ofrece enlaces directos a las webs oficiales donde puede obtener el juego, con un solo click podrá ir allí para comprarlo.

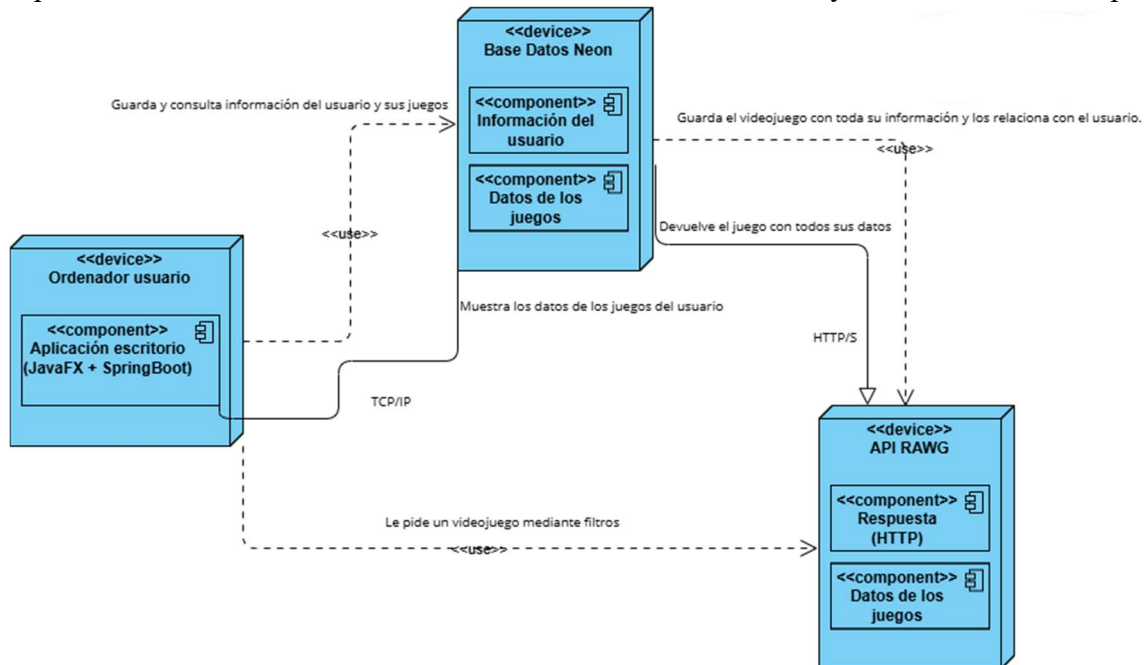
Diagrama navegación



Arquitectura

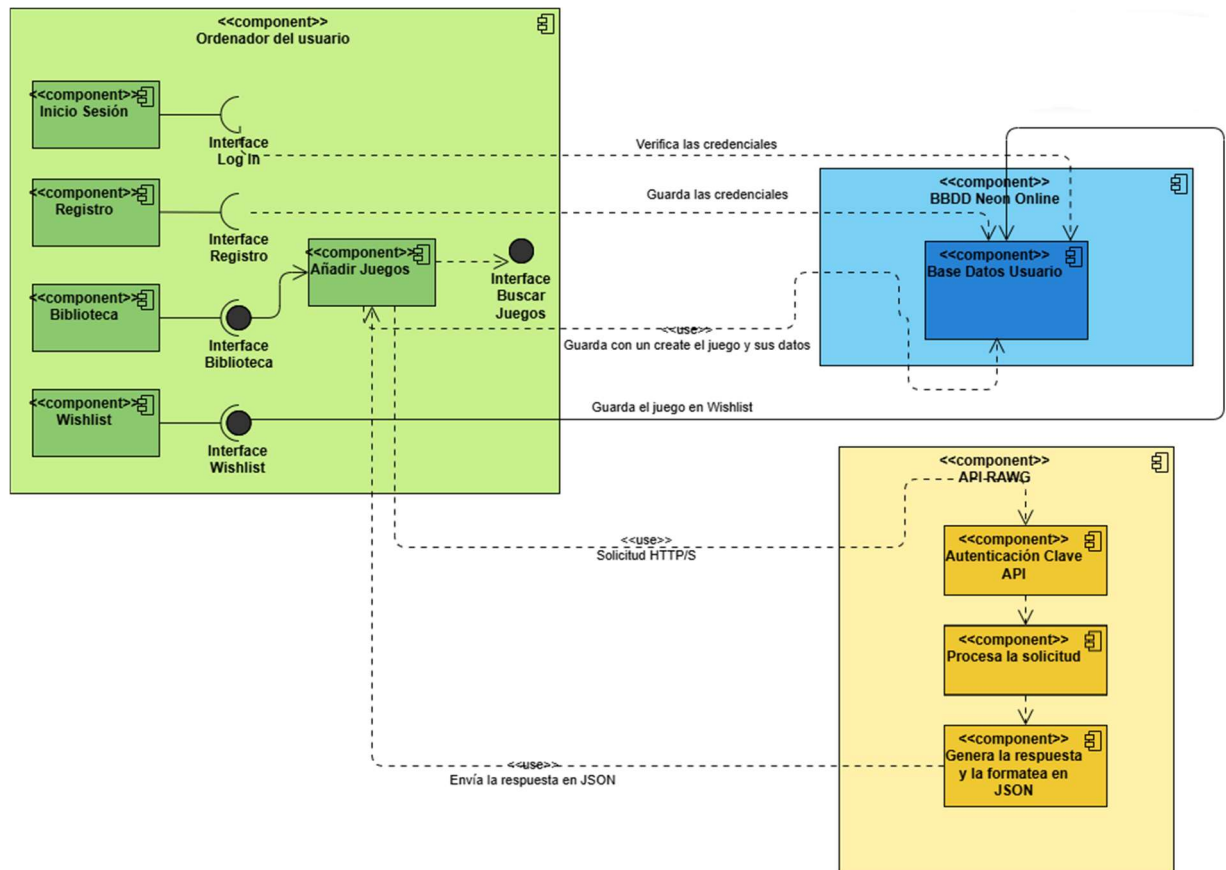
Diagrama

Aplicación de escritorio con base de datos en la nube y consultas a un api:



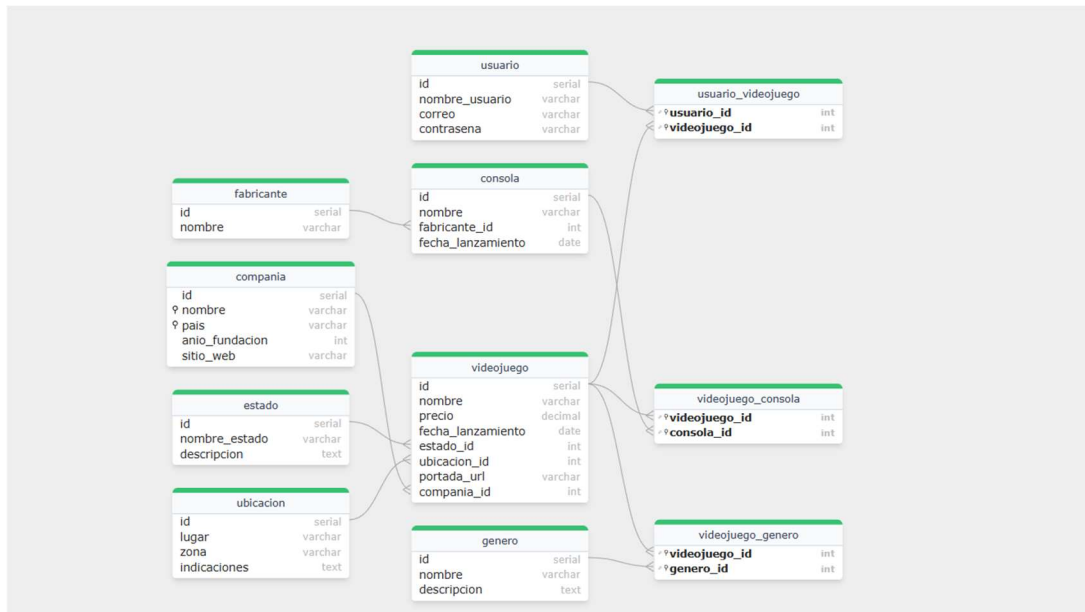
Componentes

La aplicación utiliza el modelo vista controlador, arranca con Spring Boot. Backend junto con JavaFX para las interfaces. Las consultas a la API se realizan mediante un cliente REST usa retrofit para consumir la API de RAWG, además de okhttp para manejar las respuestas de la api. Esta API proporciona información detallada y actualizada sobre videojuegos y es fundamental para llenar la base de datos del usuario sin necesidad de introducir manualmente todos los datos. Para el acceso a datos, se emplea Spring Data JPA con Hibernate. Los repositorios implementan JPA para facilitar el CRUD y reducir código. Gracias a esto se pueden abstraer las consultas SQL y aplicar buenas prácticas en el diseño, como la separación de responsabilidades entre capas. Esto proporciona una forma robusta y mantenible de integrar servicios REST. Luego, tenemos las clases que representan cada una de las tablas que están presentes en la base de datos, además de los repositorios donde se encuentran las consultas a la base de datos, y a su vez tenemos los servicios, que utilizan los métodos del repositorio, para separar responsabilidades.



Base de datos

Base de datos PostgreSQL alojada en Neon. Se utiliza Hibernate como ORM de la aplicación, además de JPA para simplificar las consultas PostgreSQL (CRUD) además de anotaciones.



Plan de pruebas

Vamos a realizar pruebas sobre las diferentes clases de servicio y otras funcionalidades, como la interacción con la API o la conexión con la misma o la base de datos en línea que almacena la información de los usuarios. Se usará Junit para realizar las pruebas junto con Jupiter y Mock para “simular” las diferentes clases u objetos que necesitaremos para el programa. Se van a realizar las siguientes pruebas unitarias:

Pruebas para el registro (Clase UsuarioServicio):

Método registrarUsuario():

- Registro válido y exitoso
- Error: Campos vacíos o nulos
- Error: Formato del e-mail no es válido
- Error: El mail o usuario ya están registrado
- Error: La contraseña no cumple los mínimos de seguridad (9 caracteres, 1 mayúscula, 1 minúscula y un carácter especial)
- Error: Las contraseñas no coinciden

Pruebas para inicio de sesión (Clase UsuarioServicio):

Método validarCredenciales():

- Login exitoso
- Error en el login: Contraseña o nombre de usuario incorrectos.

Pruebas sobre los videojuegos en la App (Clase VideojuegoServicio):

Método guardar():

- Guardar un juego con su género, consola y empresa.

- No guarda si el género o la consola no existen (se prueba por seguridad, pero tal como está programado, salvo que cambie algo en la api, es imposible)

Método obtenerTodos():

- Comprobar que devuelve una lista de todos los juegos almacenados

- Comprobar que devuelve los videojuegos por su nombre

Método obtenerPorNombreYFechaDeLanzamiento():

- Obtener un videojuego por nombre y fecha de lanzamiento (para comprobar que es único, pues un juego puede tener el mismo nombre, pero es difícil que tenga la misma fecha de lanzamiento)

Método borrar():

- Borrar un videojuego

Pruebas sobre los videojuegos de un usuario específico (Clase Usuario):

Método obtenerVideojuegosPorUsuario()

- Devolver todos los juegos de un usuario

Método tieneVideojuego()

- Comprobar si el usuario tiene un juego específico

Método agregarVideojuegoAUsuario()

- Agregar un videojuego a la lista del usuario

Método eliminarRelacionUsuarioVideojuego()

- Eliminar la relación entre el usuario y el juego (eliminarlo de la biblioteca)

Método obtenerVideojuegosEnWishlist()

- Devolver una lista con todos los videojuegos en la wishlist del usuario

Pruebas para guardar la compañía (clase CompaniaServicio):

Método guardarSiNoExiste()

- Comprobar que guarde la compañía en caso de no existir en la tabla.

- Si ya existe, devuelve la misma en lugar de guardarla

- No guarda una compañía si no obtiene el nombre

Pruebas para guardar una ubicación (clase UbicacionServicio):

Método guardarSiNoExiste()

-Si no existe, la guarda, si existe, devuelve la que ya existe.

Luego, también realizaremos pruebas no unitarias, para confirmar que las funcionalidades de la aplicación como los filtros de los videojuegos, la paginación o las conexiones funcionan:

Pruebas de los filtros para los juegos en la bbdd (clase FiltrarVideojuego):

Método filtrarYOrdenar()

-Comprobar que los filtros funcionan (por el nombre, por la consola, por el género...)

-Comprobar que la ordenación por nombre (A-Z, Z-A) y por fecha (más reciente, más antiguo) funciona

Pruebas sobre la paginación (clase Paginador):

Método getPaginaActual()

-Comprobar que controla la página actual correctamente.

Método irSiguiente()

-Comprobar que avanza de página correctamente

Método irAnterior()

-Comprobar que retrocede a una página anterior correctamente

Método estaVacio()

-Comprobar si la “lista” (de videojuegos) está vacía, por lo tanto, no habría páginas.

Pruebas de conexión (no unitarias):

-Comprobar conexión a la BBDD

Implementación

Entorno de Desarrollo

Lenguaje de programación: Java

Versión de Java: 22

IDE: Visual Studio Code

Gestor de dependencias: Maven

Sistema Gestor de Base de Datos: Neon (PostgreSQL)

Dependencias usadas: SpringBoot, Lombok, Hibernate JPA, PDMModel, Jackson, Jakarta Validation, Retrofit

Control de versiones: GitHub

API: RAWG

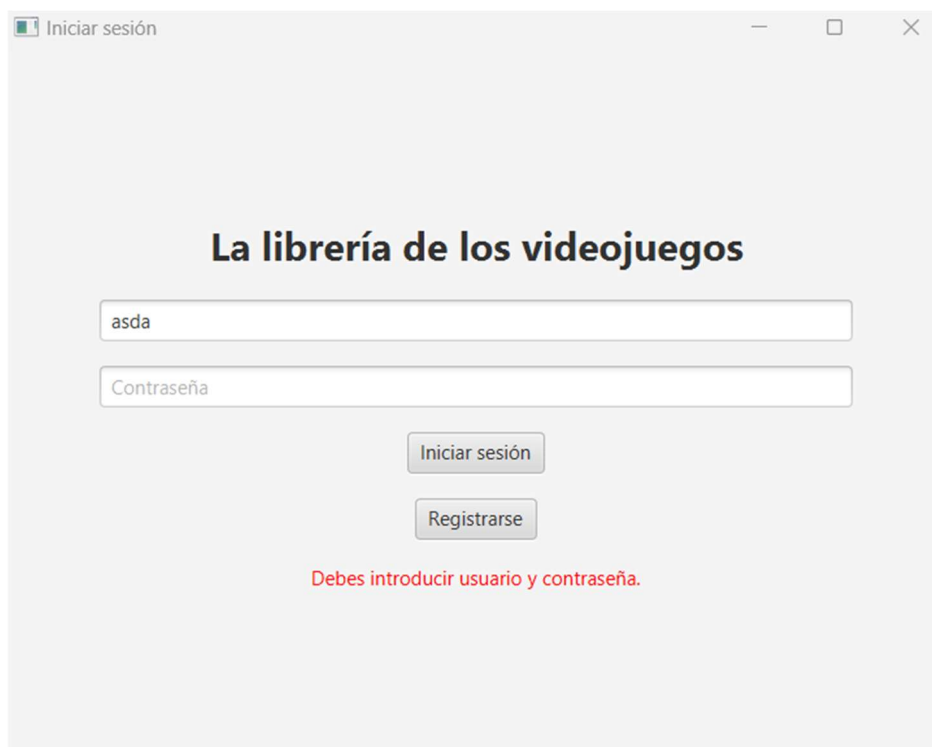
Implantación/Puesta en producción

La aplicación se ejecuta localmente mediante Maven con conexión a una base de datos alojada en la nube. Cuando se desea añadir un videojuego, se realiza una consulta a una api con videojuegos (RAWG), se puede buscar mediante nombre, consola etc. Si pulsas el botón de agregar, se agregará a tu biblioteca. También tiene un sistema de log in que permite tener varios usuarios con sus propias bibliotecas.

Capturas en ejecución de la aplicación (funcionalidades)

En este apartado se muestran las pantallas de la aplicación y cómo se comportan ante diferentes escenarios (errores, búsquedas, consultas...)

Introduciendo un usuario incorrecto:



Iniciar sesión

La librería de los videojuegos

Iniciar sesión

Registrarse

Usuario o contraseña incorrectos.

Registrando un usuario con algún dato incorrecto:

Registro de Usuario

Registro de Usuario

Todos los campos son obligatorios.

Registrarse

Volver a Iniciar Sesión

Registro de Usuario

Registro de Usuario

adasdas

sdasd

.....

.....

El correo electrónico no tiene un formato válido.

Registrarse

Volver a Iniciar Sesión

Registro de Usuario

Registro de Usuario

adasdas

mail@mail.com

.....

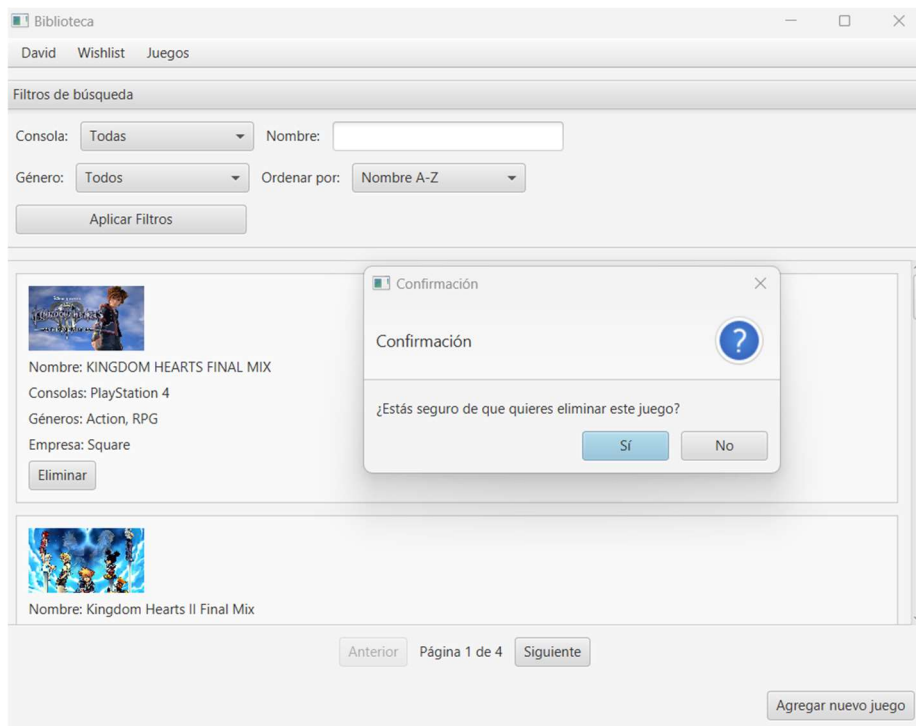
.....

La contraseña debe tener al menos 8 caracteres, una mayúscula, una minúscula, un número y un carácter especial.

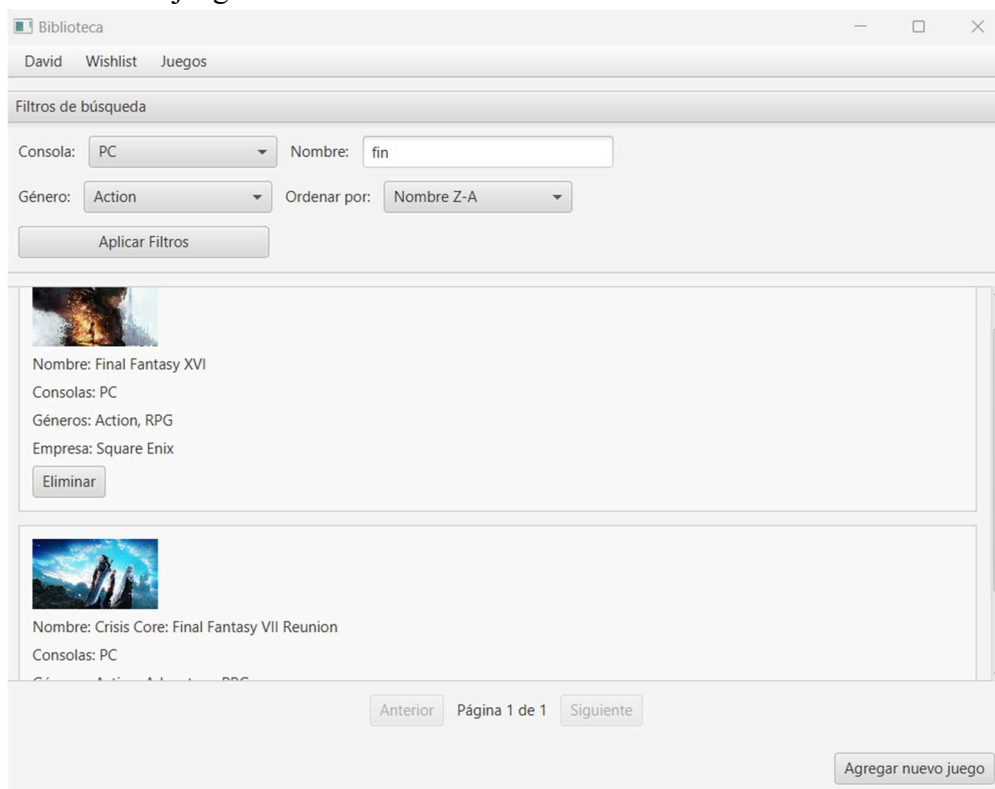
Registrarse

Volver a Iniciar Sesión

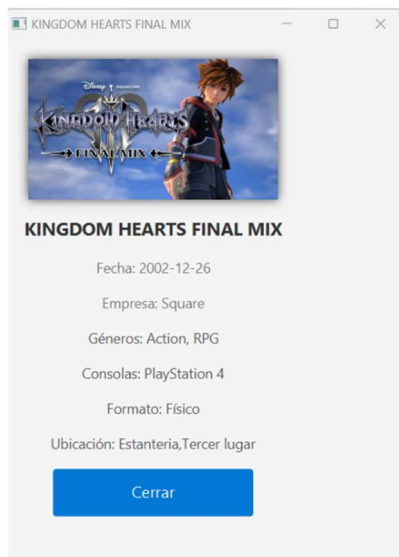
Eliminando un juego:



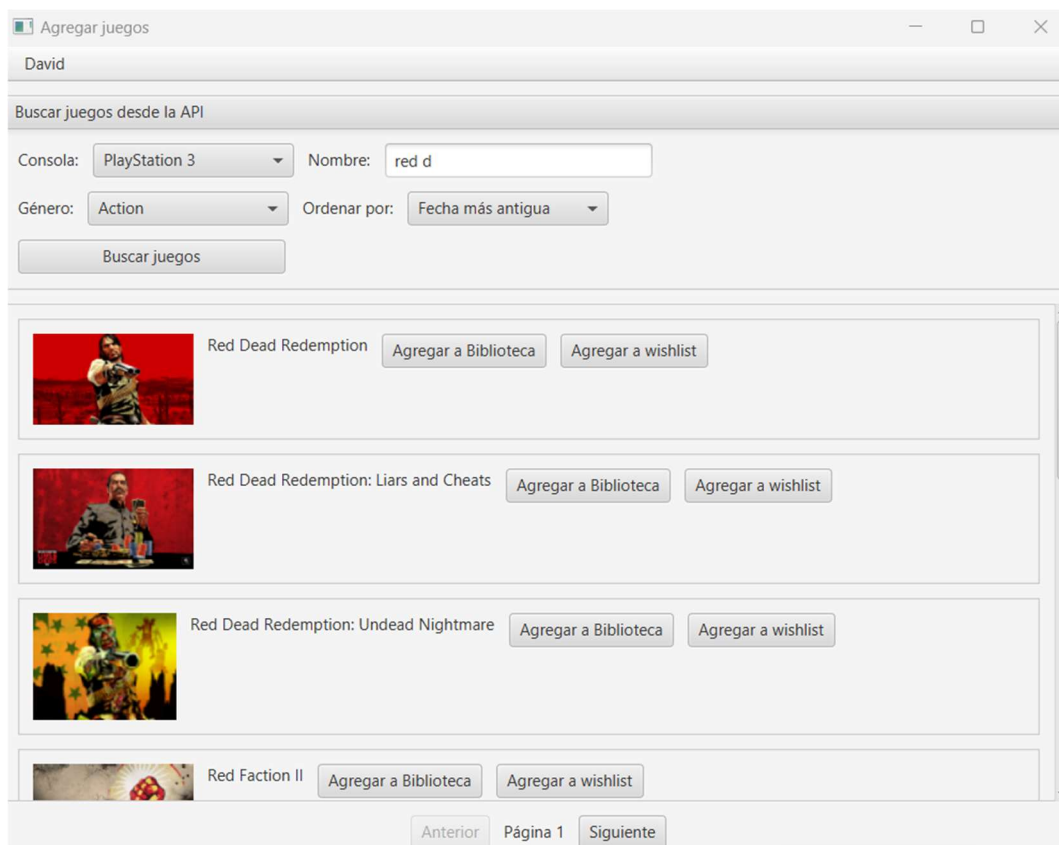
Filtrando un juego:



Abriendo la ficha de un juego:



Buscando un juego para añadir:



Añadiendo un juego a la biblioteca:

Agregar juegos

David

Buscar juegos desde la API

Consola: PlayStation 5 Nombre: expedi

Género: Todos Ordenar por: Fecha más reciente

Buscar juegos

Clair Obscur: Expedition 33

Expeditions: A MudRunner

We Were Here Expeditions

Starseeker: Astroneer Expeditions

Anterior **Página 1** Siguiente

Seleccionar formato

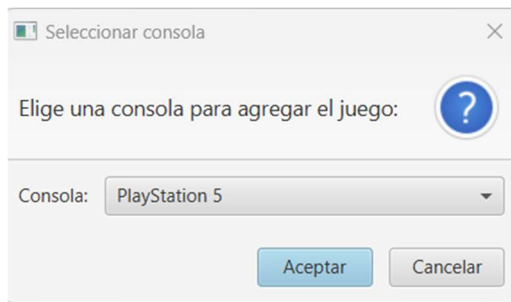
¿Cómo tienes este juego?

Formato: Físico

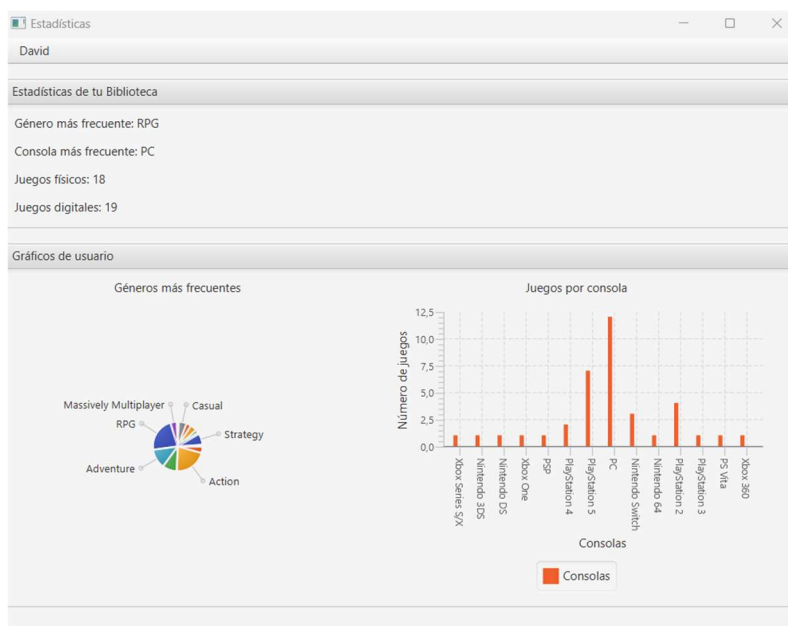
Seleccionar ubicación

Selecciona una ubicación existente o cancela para crear una nueva:

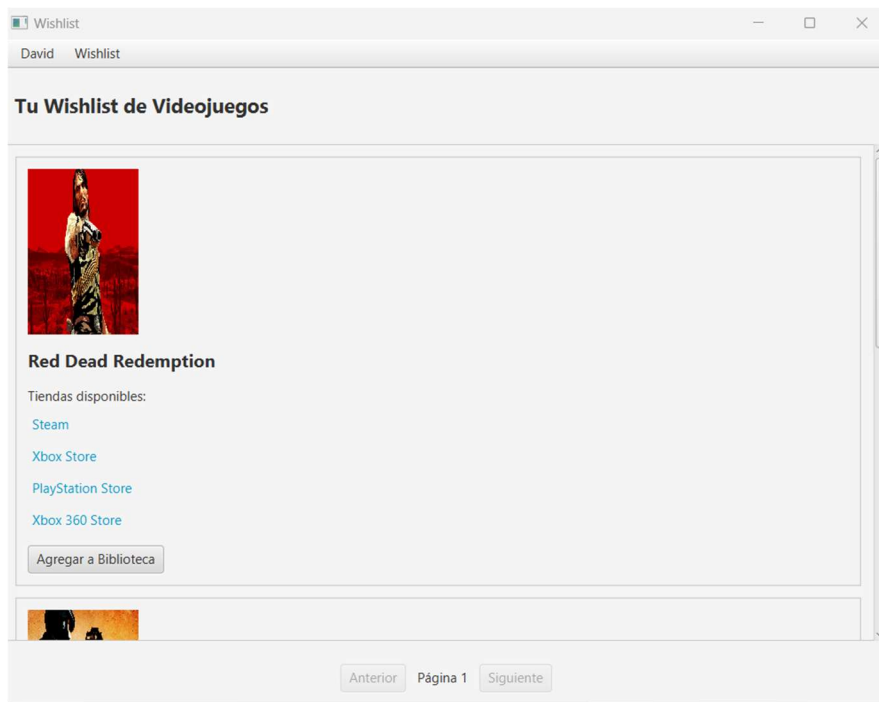
Ubicación: Ubicación(id=1, lugar=Estanteria, zona=Tercer lugar, indicaciones=En la habitación, la estantería encima de la TV.)



Mostrando estadísticas del jugador:



La wishlist del jugador:



Información sobre la versión y software necesario

- Java 22
- Visual Studio Code
- Neon DB
- Maven
- GitHub
- Scene Builder
- Conexión a Internet para uso completo

Elementos destacables del desarrollo

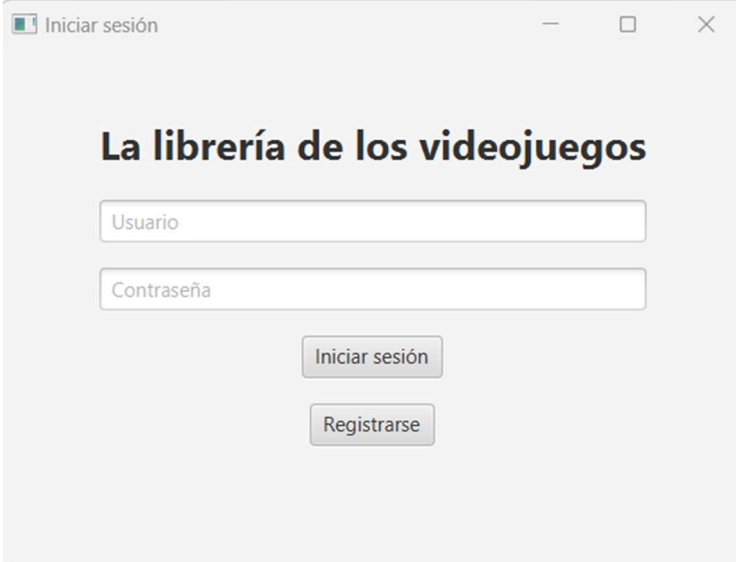
- Integración con APIs para la búsqueda de videojuegos y agregarlos en la bbdd, para no tener necesidad de crear una base de datos propia de donde añadir todos los videojuegos, si no ir rellenando nuestra base de datos extrayendo los datos de la api con los datos de los videojuegos.
- Enlace externo a las webs oficiales para comprar los videojuegos que el jugador ha añadido a su wishlist
- Base de datos en la nube para poder tener acceso a la misma en cualquier situación (siempre y cuando se tenga internet) y evitar problemáticas como perder la base de datos debido a un formateo o tener que reinstalar el SO.

- Estadísticas basadas en nuestra biblioteca y exportación de la biblioteca en formato pdf para poder mostrarla de manera sencilla.
- Uso de JavaFX moderno para poder tener unas interfaces simples pero funcionales
- Modularidad y escalabilidad en caso de querer añadir nuevas funcionalidades.

Manual de usuario

Este es el manual de usuario donde explicaremos como utilizar la aplicación con todas sus funciones. Empezaremos con las primeras funciones que verá el usuario: el log in, o el registro en caso de no tener una cuenta.

Inicio de sesión:



The screenshot shows a JavaFX window titled "Iniciar sesión" (Login). The window has a title bar with standard OS controls (minimize, maximize, close). The main content area has a title "La librería de los videojuegos" (The video game library) in a bold, dark font. Below the title, there are two text input fields: the first is labeled "Usuario" (Username) and the second is labeled "Contraseña" (Password). Below these fields, there are two buttons: "Iniciar sesión" (Login) and "Registrarse" (Register). The buttons are rectangular with rounded corners and a light gray background.

Vamos a explicar un poco cómo funciona esta pantalla. Lo primero que necesitarás será, además de conexión a internet para verificar las credenciales, una cuenta, si no la tienes, pasa al apartado de la pantalla de registro, si por el contrario ya la tienes, vamos a explicar cómo puedes iniciar sesión. Este es un proceso bastante sencillo. Simplemente tendremos que introducir nuestro nombre de usuario (recuerda que las mayúsculas y minúsculas cuentan, y los nombres de las cuentas son únicos, no habrá dos cuentas con el mismo nombre) y la contraseña asociada a dicha cuenta. Si estamos registrados y todos los datos son correctos, podremos iniciar sesión y pasar a la pantalla de biblioteca.

Registro de usuario:

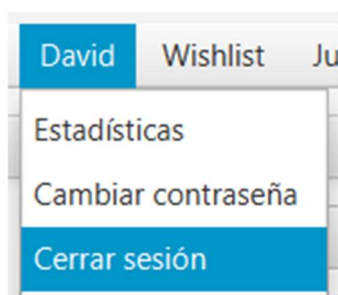
A screenshot of a web application window titled "Registro de Usuario". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The title "Registro de Usuario" is centered at the top. Below the title, there are four input fields stacked vertically: "Nombre de Usuario", "Correo electrónico", "Contraseña", and "Confirmar Contraseña". Each field has a light gray border and placeholder text. Below the input fields, there are two buttons: "Registrarse" and "Volver a Iniciar Sesión". The "Registrarse" button is a light gray rectangle, and the "Volver a Iniciar Sesión" button is a slightly lighter gray rectangle.

Si en su caso usted no tiene un usuario registrado, deberá primeramente registrar uno. Antes de nada, hay que tener en cuenta que el usuario debe ser ÚNICO. No podrás poner el mismo nombre de usuario o correo en dos usuarios diferentes. El nombre de usuario no es case sensitive, puedes tener Usuario1 y usuario1 sin problemas, el correo debe cumplir el patrón clásico de un correo (correo@mail.com) y la contraseña tendrá una restricción por temas de seguridad. Deberá contener al menos, una mayúscula, un número y un carácter que no sea una letra y tampoco uno restringido (por ejemplo, “!”, sería un carácter válido, pero el carácter “.” no) y deberá tener como poco, 8 caracteres de longitud (incluyendo los requeridos). Una vez sepas las restricciones, puede proceder a introducir los datos de tu usuario. Deberás introducir dos veces la contraseña, por temas de seguridad (si no coincide dará error el registro) y una vez tengas todo, puedes darle al botón de registrar y si todo ha salido bien, te llevará a la pantalla del inicio de sesión para entrar con tu usuario.

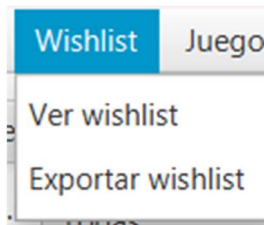
Biblioteca de juegos:

Este es el corazón de la aplicación, será el lugar principal de la misma, desde el cual podremos gestionar la biblioteca de videojuegos del usuario logueado, y además será la pantalla principal de la aplicación, desde la cual podremos ir a las otras, o realizar diferentes acciones.

Vamos a empezar de arriba abajo:



Aquí podemos ver el nombre del usuario logueado (en este caso David) y las diferentes cosas que podemos hacer desde aquí: Está Estadísticas, que nos llevará a la pantalla de las estadísticas del usuario logueado, cerrar sesión, que cerrará la sesión actual, llevando al usuario a la pantalla de log in para iniciar sesión de nuevo, y cambiar contraseña, para como dice su nombre, cambiar su contraseña (cambiar contraseña no es funcional)



Luego, al lado de esta opción, tenemos la de la wishlist, desde donde podremos tanto ir a la misma como exportarla en formato PDF. No hay mucho más que comentar en este apartado, lo vamos a comentar más en profundidad en el siguiente punto, ya que tanto el de wishlist como el de juego son iguales en cuanto a exportación.

Por último, en lo referente al menú superior, tenemos en juegos, la opción de exportar la biblioteca en formato PDF. En este punto nos vamos a parar para explicar cómo hacerlo y el resultado que deberíamos obtener: Aunque dijimos en el anterior apartado que aquí lo explicaríamos más en profundidad, realmente no tiene tampoco misterio. Simplemente debes pulsar en juego y se desplegará un sub menú como el de la foto, solamente habrá una opción que es exportar biblioteca, la pulsas y se te abrirá el explorador de archivos (Windows) para guardar el documento .pdf con los datos de la biblioteca, además de darle un nombre al mismo. Cuando le pongas un nombre y selecciones una ubicación, el archivo se descargará en dicha ubicación, y debería quedar algo así:

Biblioteca de Juegos de David

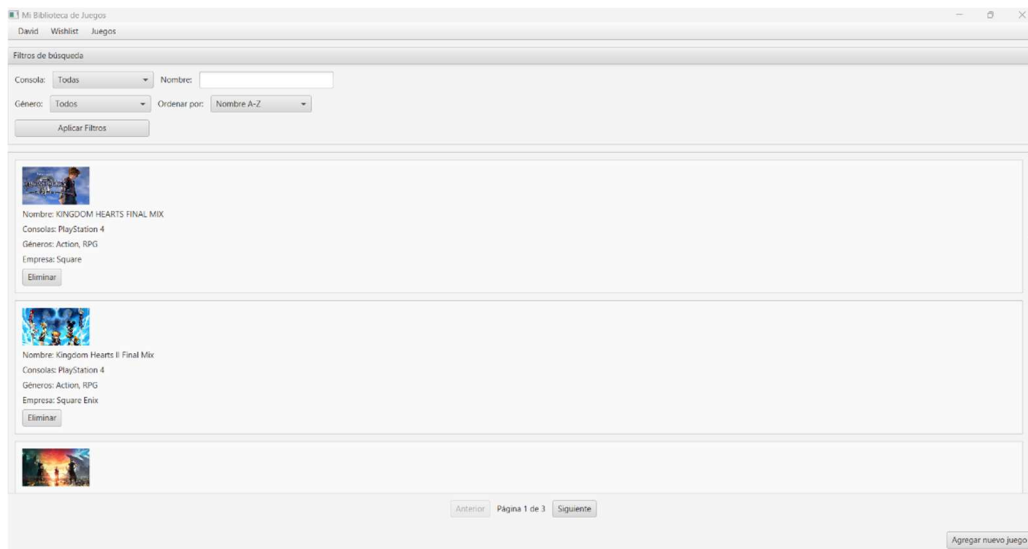
Nombre: KINGDOM HEARTS FINAL MIX
Consolas: PlayStation 4
Géneros: Action, RPG
Empresa: Square

Nombre: Kingdom Hearts II Final Mix
Consolas: PlayStation 4
Géneros: Action, RPG
Empresa: Square Enix

Nombre: Final Fantasy VII Rebirth
Consolas: PlayStation 5
Géneros: Action, Adventure, RPG
Empresa: Square Enix

El formato sería: añadir al título el nombre del usuario logueado que está exportando su biblioteca, debajo de este, los diferentes juegos con su título, consola, géneros y empresa (desarrollador). Así para todos los juegos.

Vamos ahora con los filtros y la biblioteca en general, para ello, vamos a mostrar una imagen de la pantalla entera para observar las diferentes opciones:



Lo primero que vemos son los filtros disponibles: Podemos ver que hay un desplegable con las consolas, donde al pulsarlo nos aparecerán todas las consolas que devuelve la api de RAWG, o lo que es lo mismo, las consolas de las que podremos añadir juegos en nuestra aplicación. La opción por defecto es todas, lo cual significa que no filtrará por consolas y mostrará todas.

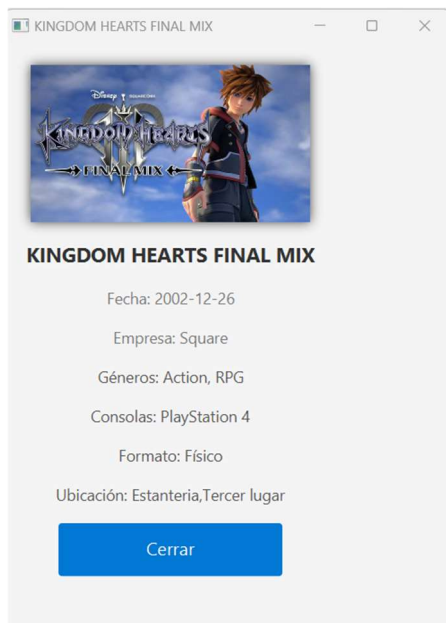
La siguiente opción que tenemos son los géneros, al igual que las consolas, es un desplegable con todos los géneros que ofrece RAWG. De nuevo está la opción

El último desplegable que tenemos será el de ordenar por, el cual ofrece ordenar de la a a la z, en inversa, por fecha más reciente, fecha más antigua o ninguno.

Luego tenemos el campo de texto donde podemos introducir el nombre del juego. En el tema filtros, siempre priorizará el nombre, mostrando primero los juegos que coincidan exactamente con lo escrito, o los que más se parezcan, aunque ten en cuenta que si por ejemplo buscas “Pokemon” y en consolas seleccionas “PlayStation5” o en género seleccionas “Juego de Mesa” no te saldrá, básicamente los filtros se compenetran y sincronizan entre si para no dar incongruencias a la hora del filtrado.

Una vez aplicados los filtros deseados, se pulsará aplicar filtros y se mostrarán únicamente los videojuegos que estén en la tabla del usuario logueado, y que coincidan con los o el filtro seleccionado. Por defecto se cargará la biblioteca entera con una paginación que muestra 10 juegos por página.

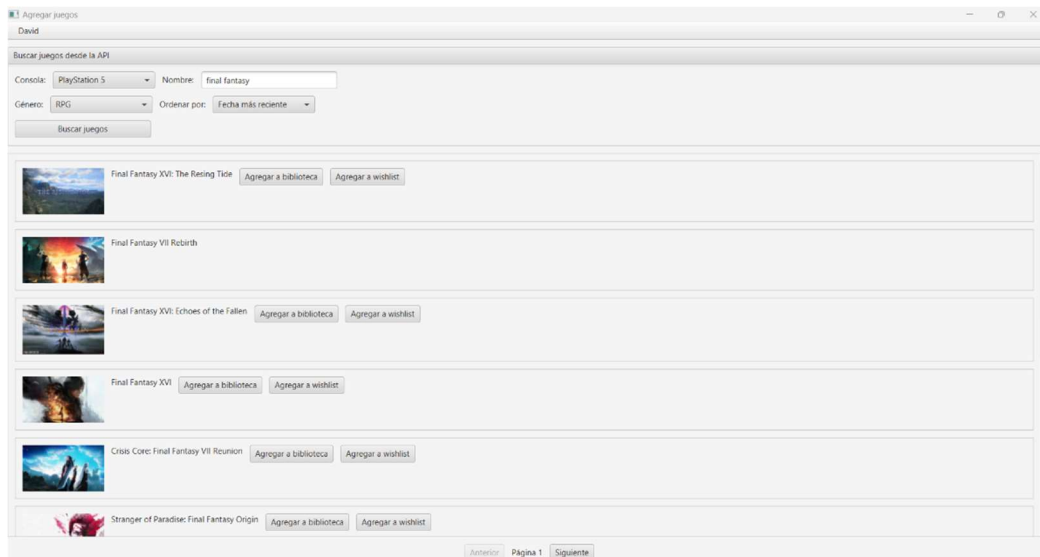
En las fichas de los juegos, nos sale una “preview” con algunos datos del juego, pero si pulsamos uno de los juegos, se nos abrirá en otra ventana una ficha de dicho juego con todos los datos (que tenemos guardados) de dicho juego:



Debajo de la biblioteca, tenemos los botones para pasar de página en caso de tener más de 10 juegos.

Por último, para ir a añadir un juego a la biblioteca, tenemos la opción de agregar nuevo juego, que nos abrirá la pantalla de búsqueda, desde la que realizaremos las consultas a la api utilizando filtros similares. Ese será el siguiente punto que trataremos:

Búsqueda de juegos para añadir:

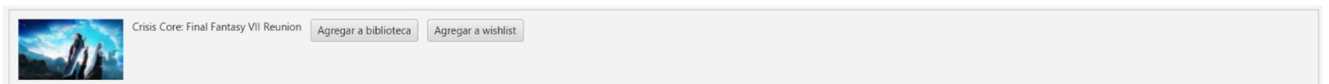


Esta pantalla se podría decir que es el “segundo motor” de la aplicación, ya que si la de biblioteca es el corazón y el núcleo desde el cual puedes ir a cualquier otra pantalla, esta es la otra que es clave, pues es desde donde obtendremos los juegos para llenar la propia biblioteca, ver las estadísticas del usuario, o poder exportar la biblioteca. Básicamente, sin estas dos, la aplicación no podría funcionar.

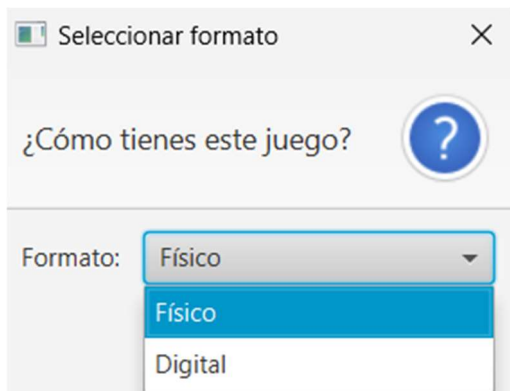
Lo primero que vemos son filtros similares a los de la biblioteca, podemos abrir desplegables para añadir y filtrar por una consola, por un género, ordenar esos resultados por fecha de salida más reciente o más antigua, y luego el campo de texto para añadir el nombre. De nuevo, priorizará el nombre, mostrando los que más se asemejen a lo que escribimos (si escribimos algo), igual que con la biblioteca, solo que aquí en lugar de a la biblioteca del usuario logueado, hacemos la consulta a la api RAWG. Cuando tengamos todos los filtros que queremos añadir, podemos darle a buscar.

Abajo nos salen los juegos que se han encontrado, en un formato similar a la biblioteca, pero sin la preview de los datos del juego, solo aparecerá la portada del juego junto al nombre del mismo, además de dos botones: Uno para añadir el juego a la biblioteca, y otro para añadirlo a la wishlist, salvo que dicho juego ya esté en la biblioteca, en cuyo caso no saldrá ningún botón, y en caso de estar en la wishlist, solo aparecerá el de añadirlo a la biblioteca (entendiendo que ya lo has comprado). Aquí vamos a pararnos porque tenemos que explicar cómo añadir un juego:

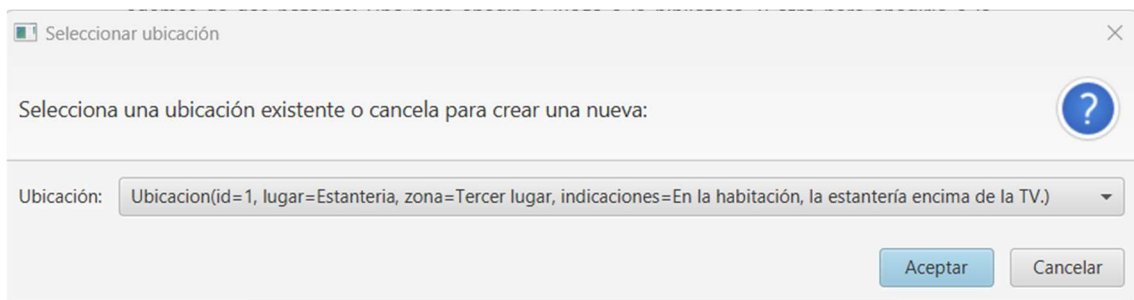
Siguiendo la búsqueda de la foto anterior, vamos a agregar el Crisis Core Final Fantasy VII Reunion



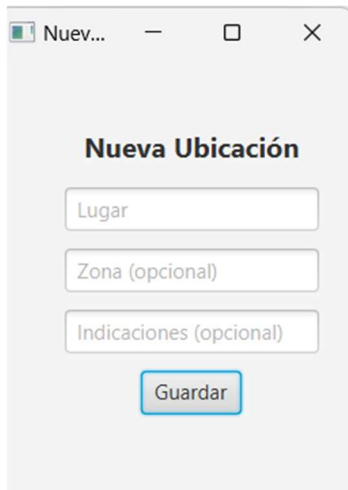
Pulsamos el botón agregar a biblioteca, y entonces nos preguntará algunas cosas:



La primera será el formato, nos preguntará si es físico o digital. Si le decimos digital, pasará a la siguiente, pero si le decimos físico, nos hará una pregunta extra. Para mostrar todo, vamos a seleccionar físico:



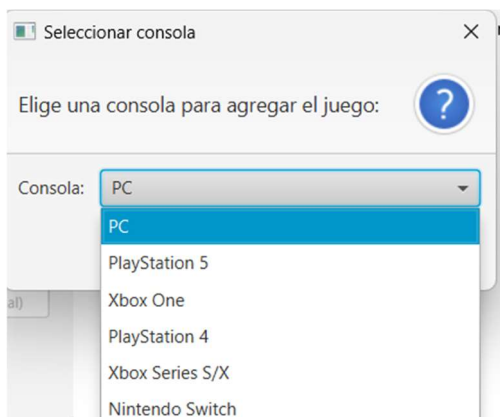
Nos preguntará la ubicación del juego. Tenemos dos opciones: Usar una ubicación ya registrada, o crear una nueva (en caso de no tener ninguna guardada, deberás crearla). Vamos a elegir crearla nosotros para mostrarlo, para ello, como dice el texto, debemos pulsar cancelar:



Formulario de "Nueva Ubicación" con los siguientes campos:

- Lugar
- Zona (opcional)
- Indicaciones (opcional)
- Botón Guardar

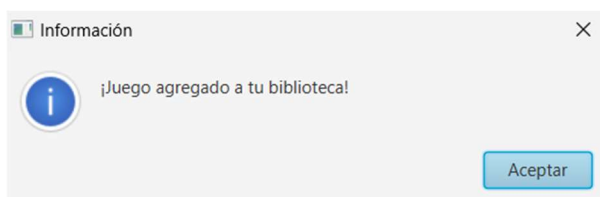
Entonces se nos abrirá un formulario para agregar la zona, donde debemos agregar el lugar donde está (una estantería, un armario...) y como indicaciones opcionales, podemos añadir la zona (segundo estante, parte superior...) y unas indicaciones (dónde se encuentra específicamente, en una fila de juegos, el tercero por la derecha, está en un trastero, en una habitación específica...) Cuando tengamos todo, podemos darle a guardar.



Formulario de "Seleccionar consola" con el mensaje: "Elige una consola para agregar el juego:". El desplegable de consolas muestra:

- PC
- PlayStation 5
- Xbox One
- PlayStation 4
- Xbox Series S/X
- Nintendo Switch

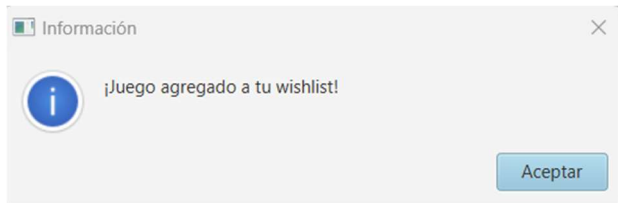
Lo siguiente que nos preguntará, en caso de estar disponible para más de una consola, es para qué consola lo queremos guardar. Abrimos el desplegable y seleccionamos la consola. Una vez la tengamos le damos a aceptar, y nos saldrá un mensaje como este:



Mensaje de confirmación: "¡Juego agregado a tu biblioteca!". Botón Aceptar.

Eso significará que ya tenemos el juego en nuestra biblioteca agregado.

Para agregar a la wishlist es más sencillo, solamente debemos pulsar el botón, y ya se agregará, no nos pide ningún dato, puesto que es solo una wishlist, saldrá un mensaje como este:



De esta pantalla solamente nos queda comentar la paginación, que al igual que la biblioteca, carga 10 juegos por página, y en este caso, tendrá tantas páginas, como resultados devuelva el api según los filtros que le hemos dado.

Debemos comentar, que si un juego desde esta pantalla, que ya has agregado a la biblioteca, lo agregar también a la wishlist, se borrará de la biblioteca, tendrás que ir a la wishlist a agregarlo a la biblioteca, o volver a realizar la búsqueda en el controlador. Normalmente harás lo primero por comodidad. Para volver a la pantalla inicial, tienes que pulsar en el menú superior donde pone el nombre del usuario y te saldrá la opción.

Hablando de la wishlist, vamos ahora con dicha pantalla:

Wishlist:

Vamos ahora con la wishlist:



Es una pantalla mucho más minimalista que las anteriores, aunque mantiene la estética de toda la aplicación. No tenemos los filtros, pues no es necesario para la wishlist. Vemos que lo que muestra de cada juego es una foto más las tiendas oficiales donde se pueden comprar (las que ofrece la api de RAWG al menos) que si las pulsamos, nos llevará a las mismas para poder comprar el juego. Además de una paginación igual a las vistas anteriormente (aunque en este caso solamente muestra 5 juegos por página). Pulsando arriba, en el menú donde pone el nombre del usuario, podemos volver a la biblioteca.

Vamos ahora con la pantalla de las estadísticas:

Estadísticas:



Antes de empezar, para aclarar, estas estadísticas SOLO mostrarán los juegos que estén en la biblioteca del usuario. No van a tener en cuenta los juegos que estén en la wishlist obviamente, pues no son juegos que el usuario tenga.

Podemos ver las estadísticas resumidas en la primera línea: El género favorito (o del que más juegos tenemos), lo mismo con la consola, además del total de juegos digitales y físicos. Abajo tenemos las estadísticas de géneros y consolas más bonitas en un gráfico de tarta y otro de los juegos por consola en un gráfico de barras. Esto se hace sobre todo aparte de ofrecer algo más visual, porque puede haber muchos juegos de muchas consolas o muchos juegos de muchos géneros, a diferencia por ejemplo de los juegos físicos o digitales, donde solo hay dos opciones.

Si deseas volver a la biblioteca, debes pulsar en el menú de arriba el nombre de la cuenta y darle a la opción de volver a la biblioteca.

Conclusiones

Ha sido un desarrollo bastante complicado comparado con los realizados en clases o exámenes. Es cuándo más he tenido que emplear de ayudas externas, ya sea vídeos de youtube, preguntar a personas con conocimiento en el tema, pero creo que al final, sobre todo en la parte final, que era la que suponía iba a llevar más problemas (conectar con la api, recibir la respuesta, formatearla...) ha sido, dentro de sus complicaciones como todo, más sencillo de lo esperado, al menos si lo comparamos con la parte inicial, tanto la previa a la programación, como los diseños de las interfaces, los diferentes diagramas previos, y la creación del esqueleto del programa ya en código, no ha sido tan complicado. La mayor dificultad fue a la hora de mostrar usar javaFX para mostrar las interfaces junto a SpringBoot, ya que había muchos errores tontos pero que te atascan durante horas (como un atributo del fxml que en el controlador tenía una anotación Transactional que evitaba que se ejecutase el programa). Luego hubo más problemas, como una necesidad de reestructuración del diseño a mitad del desarrollo, por la forma en la que devuelve la api de RAWG consolas y géneros, pero nada grave ni difícil, salvo lo engorroso de tener que

realizar alter tables, cambiar la estructura de las entidades en código, y por suerte esto fue antes de codificar la parte de las consolas y géneros, así que código no fue necesario modificar. Luego algún problema con el responsive y el manejo de scene builder y java fx al inicio como ya comenté, más por falta de experiencia que otra cosa. Al final del día, pienso que mi aplicación actualmente, salvo uno o dos detalles que la diferencia del resto, los cuales ya explico en este documento, no es la gran revolución, pero creo que como poco, ofrece algo nuevo que otras aplicaciones de este tipo no ofrecen, en un formato diferente, y honestamente pienso que es una aplicación que yo podría usar en caso de querer llevar un control más exhaustivo de toda mi colección de videojuegos, para tener control de todos, sobre todo los físicos en caso de mudanzas, obras o cambios de ubicación. Por supuesto que había más cosas que me hubiese encantado implementar, y cosas implementadas que me hubiese gustado mejorar, pero preferí no apurar y dejar una semana y algo de margen para pulir el documento y realizar las pruebas unitarias, aunque ya he probado yo todas las funcionalidades según las iba implementado. Creo que también ha sido un aviso importante de cara al futuro laboral, para saber que con lo aprendido durante el ciclo no es suficiente, y que cuando empieces en una empresa, vas a ir prácticamente de cero, y sobre todo, intentar cada vez depender menos de ayudas externas, e ir aprendiendo en lugar de tener que andar preguntando o consultando vídeos o diferentes inteligencias artificiales, igualmente, cuánto más avanzaba el proyecto, más he ido mejorando en este sentido, aunque me hubiese gustado ser más independiente en ese sentido, más que nada como dije, de cara al mundo laboral, donde por supuesto, todo esto son herramientas, pero también hay que tener cuidado de no depender mucho de ellas y aprender de las dudas que te resuelven y que se te quede para la próxima que tengas un error similar. Otros problemas que me he ido encontrando ha sido el uso conjunto de javafx y spring boot, pues spring boot como ya se ha comentado, está más orientado a aplicaciones web, por lo tanto, no tiene mucha compatibilidad con las interfaces, ni con java fx. Tuve que estar casi 1 semana, de las primeras del desarrollo, para solucionar y que cargasen todas las pantallas. La mayoría de veces eran errores de compatibilidad con el “Bean” de spring boot y los fxml, en la forma en la que los interpretaba cada uno, al final tuve que separar java fx y spring boot para encontrar el error tras muchas consultas y pruebas. Desde entonces, no me ha vuelto a dar error, salvo algunas veces por temas de incompatibilidad de algunos elementos del scene builder por algún motivo que no me ha quedado muy claro con spring boot, pero con modificarlos o quitarlos se solucionaba. En el tema de la base de datos, hubo un error que por suerte se detectó al inicio del desarrollo, que es que al principio se usaba Supabase, el cual no permite (de forma gratuita) una conexión directa con el programa, si no que necesitaba usar peticiones http/s (como si fuese una api) lo cual no suponía mayor problema que algo más de engorro y tiempo para conectar con la BBDD, pero al hacerlo, descubrí que recientemente (entre que se realizó el anteproyecto y se llegó a este punto del desarrollo) cambiaron a una funcionalidad de pago las conexiones IPV4, permitiendo solamente IPV6, la cual no está configurada en mi red local, por lo que decidí cambiar a Neon en lugar de configurar una IPV6, que además debería seguir siendo mediante peticiones http/s, mientras que neon se conecta con IPV4, de manera directa y en la versión gratuita, solo que tiene menos capacidad de almacenamiento, pero para el tfg me daba de sobra con la que tenía. También fue muy

difícil la implementación de los filtros, ya que quería unos filtros, dentro de mis posibilidades, lo más detallados y realistas posibles, conseguir en la pantalla de búsqueda, que una consulta paginada, muestre de primero los resultados que coincidan más con el nombre escrito (si escribes fina, saldrían arriba final fantasies o juegos que tengan fina al inicio) fue más difícil de lo que parece, cuando es una funcionalidad que se da por hecho como obvia y se considera fácil, es probablemente el filtro más difícil porque es el que más quise pulir, pues al final el resto es filtrar por un género, una consola, u ordenar por diferentes patrones sencillos como la fecha de lanzamiento. La pantalla de estadísticas también fue un reto, pues pensaba hacer solamente las letras con las estadísticas, pero se me quedaba una pantalla muy sosa y vacía, y tampoco se me ocurrían más estadísticas que pudiesen ser útiles con las funcionalidades que tiene la aplicación (al principio estaba el precio, pero se tuvo que descartar, al igual que lo de juegos completados empezados, ya que al final se le ha dado un sentido más de “almacén” de juegos) y se me ocurrió meter algunos gráficos que quedasen bonitos y ocupasen esa parte que quedaba más vacía. Fue más difícil el poner en scene builder bien las gráficas casi que conectar las mismas a la bbdd y llenarlas en base a los datos, ya que esto al final es un código que puedes ver en otro lado y usar de referencia para saber cómo va. Por último, en lo referente a las pantallas, me gustaría mencionar la wishlist, que en un inicio pensaba que fuese algo más completo, pero cuando no pude implementar la api de precios, pues comprobé que rawg no tenía una función de precios, además que la api que iba a usar era de pago, y las demás no me convencieron, quedó algo desangelada. Por suerte pude incluir lo de los enlaces a las tiendas oficiales. No es perfecto, porque son solo tiendas digitales, pero al final depende de lo que devuelve la api y lo que tiene almacenado, es algo incompleto por el tema de juegos físicos, pero tampoco me parecía necesario poner por ejemplo yo los enlaces a 2-3 tiendas que vendan juegos, porque además de no aportar nada en cuanto a dificultad e implementación, tampoco sé si esos juegos están disponibles en esas tiendas (puede haber juegos retro que no estén en amazon o en game y solo en subastas de ebay por ejemplo) así que aunque no ha quedado como me gustaría, y en un futuro, si siguiese con esto, me gustaría implementar mejor el tema de tiendas y añadir lo de los precios, creo que ha quedado una pantalla útil y que no es solo “relleno”.

Bibliografía

- OpenAI. (Consultado múltiples veces entre Abril y Mayo de 2025.). *ChatGPT*. Para consultas de errores y código básico Recuperado de <https://chatgpt.com/>
- Gemini. (Consultado múltiples veces entre Abril y Mayo de 2025). *Gemini AI*. Para consultas de errores, usabilidad de interfaces y código básico Recuperado de <https://gemini.google.com/>
- Neon. (Consultado el 15 de Abril de 2025). *Neon Docs*. Para consultas de cómo conectar la app a su BBDD y realizar operaciones Recuperado de <https://neon.tech/docs>
- Oracle. (Consultado el 15 de Abril de 2025). *Java Documentation*. Para dudas básicas de programación en java Recuperado de <https://docs.oracle.com/en/java/>
- RAWG. (Consultado en Abril de 2025). *RAWG Video Games Database API*. Para consultar lo que ofrece la aplicación en cuanto a datos y cómo usarla Recuperado de <https://rawg.io/apidocs>
- Visual Paradigm. (Consultado el 10 de Abril de 2025). *Visual Paradigm Diagram Maker*. Para la creación de diversos diagramas. Recuperado de <https://www.visual-paradigm.com/>
- PlantText. (Consultado el 11 de Abril de 2025). *PlantText*. Para la creación de los casos de uso utilizando código. Recuperado de <https://www.planttext.com/>
- PlantUML. (Consultado el 18 de Mayo de 2025). *PlantUML*. Para la creación del uml de clases y paquetes. Recuperado de <https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa700001>
- Plutora. (Consultado el 30 de Mayo de 2025). *Plutora*. Web consultada para mejorar y modificar el diagrama de despliegue. Recuperado de <https://www.plutora.com/blog/deployment-diagrams-explained-in-detail-with-examples>
- Photoshop CS6. (Consultado el 11 de Abril de 2025). *Photoshop*. Para la creación de los bocetos iniciales para la creación de interfaces. Recuperado de Aplicación de escritorio.

Anexos

Repositorio del programa: `git clone https://github.com/DavidBargallo/TFG`

Para ejecutar el programa:

-Usar en la terminal del IDE el siguiente comando: `mvn javafx:run`

-Iniciar sesión con uno de estos dos usuarios:

Nombre usuario: David

Password: Contraseña3!

Nombre usuario: Test

Password: Contraseña4!

o registrar un usuario (comprobar manual).